Corresponding to US 6,317,115 B1 and US 6,693,635 B1

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平8－138033

(43)公開日　平成８年(1996) ５月31日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｔ　1/00				
Ｇ０６Ｆ　13/00	３５７　Ｚ	7368－5E		
		9365－5H	Ｇ０６Ｆ　15/ 62	

審査請求　未請求　請求項の数34　ＯＬ　（全 22 頁）

(21)出願番号　特願平6－280622

(22)出願日　平成６年(1994)11月15日

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目30番２号

(72)発明者　横溝　良和
東京都大田区下丸子３丁目30番２号キヤノン株式会社内

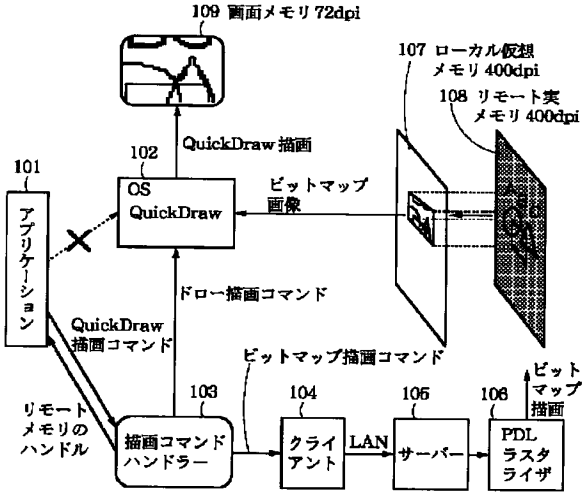
(74)代理人　弁理士　丸島　儀一

(54)【発明の名称】　画像編集装置とその方法及び画像表示制御装置とその方法及び画像編集装置と画像表示制御装置から構成されるシステム

(57)【要約】　（修正有）

【目的】　クライアントの画像描画処理をサーバーに代行させることができる。

【構成】　アプリケーション１０１からオペレーティングシステム１０２に対して発行される描画関数コールを捉えてサーバー１０５に前記描画関数コールに対応するスクリプトを発行し、該スクリプトを解釈して前記サーバー１０５上に確保されたリモート実メモリ１０８上に画像を描画し、該描画された描画画像データをネットワークを介して前記リモート実メモリ１０８から前記オペレーティングシステムに取り込み、画面メモリ１０９に表示する構成を特徴とする。



1

【特許請求の範囲】

【請求項１】　画像処理を実行するサーバーと接続された画像表示制御装置であって、
前記サーバに描画命令を出力する出力手段と、
前記出力手段で出力した前記描画命令を解釈して、前記サーバー上に確保された画像描画メモリ上に描画された画像データを取得する取得手段と、
前記取得手段で取得した前記画像データを画面メモリに表示する表示制御手段とを有することを特徴とする画像表示制御装置。

【請求項２】　前記サーバー上に確保された前記画像描画メモリは、前記画像表示制御装置の前記画面メモリより大きいことを特徴とする請求項１記載の画像表示制御装置。

【請求項３】　前記サーバー上に確保された前記画像描画メモリの解像度は、前記画像表示制御装置の前記画面メモリの解像度より高いことを特徴とする請求項１記載の画像表示制御装置。

【請求項４】　前記画像表示制御装置の画面メモリに表示される画像データは、前記サーバー上に確保された前記画像描画メモリに描画される画像データの一部であることを特徴とする請求項１記載の画像表示制御装置。

【請求項５】　前記画像表示制御装置の画面メモリに表示される画像データをスクロールすると、前記サーバー上に確保された前記画像描画メモリに描画される画像データの一部が前記画面メモリに複写されることを特徴とする請求項１記載の画像表示制御装置。

【請求項６】　前記画像表示制御装置は、ホストコンピュータであることを特徴とする請求項１記載の画像表示制御装置。

【請求項７】　前記サーバーと前記画像表示制御装置は、ネットワークを介して接続されていることを特徴とする請求項１記載の画像表示制御装置。

【請求項８】　ホストコンピュータと接続された画像編集装置であって、
前記ホストコンピュータから描画命令を入力する入力手段と、
前記入力手段で入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画する描画手段と、
前記描画手段で描画した前記画像データを前記ホストコンピュータの画面メモリに表示する為に、前記画像データを前記ホストコンピュータへ出力する出力手段とを有することを特徴とする画像編集装置。

【請求項９】　前記画像編集装置に確保された前記画像描画メモリは、前記ホストコンピュータの前記画面メモリより大きいことを特徴とする請求項８記載の画像編集装置。

【請求項１０】　前記画像編集装置に確保された前記画像描画メモリの解像度は、前記ホストコンピュータの前記画面メモリの解像度より高いことを特徴とする請求項８記載の画像編集装置。

2

【請求項１１】　前記ホストコンピュータの画面メモリに表示される画像データは、前記画像編集装置に確保された前記画像描画メモリに描画される画像データの一部であることを特徴とする請求項８記載の画像編集装置。

【請求項１２】　前記ホストコンピュータの画面メモリに表示される画像データをスクロールすると、前記画像編集装置に確保された前記画像描画メモリに描画される画像データの一部が前記画面メモリに複写されることを特徴とする請求項８記載の画像編集装置。

【請求項１３】　前記画像編集装置は、サーバーであることを特徴とする請求項８記載の画像編集装置。

【請求項１４】　前記ホストコンピュータと前記画像編集装置は、ネットワークを介して接続されていることを特徴とする請求項８記載の画像編集装置。

【請求項１５】　画像処理を実行するサーバーとホストコンピュータとがネットワークに接続されたシステムであって、
前記ホストコンピュータは、
前記サーバに描画命令を出力する出力手段と、
前記出力手段で出力した前記描画命令を解釈して、前記サーバー上に確保された画像描画メモリ上に描画された画像データを取得する取得手段と、
前記取得手段で取得した前記画像データを画面メモリに表示する表示制御手段とを有し、
前記サーバーは、
前記ホストコンピュータから描画命令を入力する入力手段と、
前記入力手段で入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画する描画手段と、
前記描画手段で描画した前記画像データを前記ホストコンピュータの画面メモリに表示する為に、前記画像データを前記ホストコンピュータへ転送する転送手段とを有することを特徴とする。

【請求項１６】　前記サーバー上に確保された前記画像描画メモリは、前記ホストコンピュータの前記画面メモリより大きいことを特徴とする請求項１５記載のシステム。

【請求項１７】　前記サーバー上に確保された前記画像描画メモリの解像度は、前記ホストコンピュータの前記画面メモリの解像度より高いことを特徴とする請求項１５記載のシステム。

【請求項１８】　前記ホストコンピュータの画面メモリに表示される画像データは、前記サーバー上に確保された前記画像描画メモリに描画される画像データの一部であることを特徴とする請求項１５記載のシステム。

【請求項１９】　前記ホストコンピュータの画面メモリに表示される画像データをスクロールすると、前記サー

バー上に確保された前記画像描画メモリに描画される画像データの一部が前記画面メモリに複写されることを特徴とする請求項１５記載のシステム。

【請求項２０】　前記サーバーと複数のホストコンピュータが前記ネットワークに接続されていることを特徴とする請求項１５記載のシステム。

【請求項２１】　画像処理を実行するサーバーと接続された画像表示制御装置における画像表示制御方法であって、

前記サーバに描画命令を出力する出力工程と、

前記出力工程で出力した前記描画命令を解釈して、前記サーバー上に確保された画像描画メモリ上に描画された画像データを取得する取得工程と、

前記取得工程で取得した前記画像データを画面メモリに表示する表示制御工程とを有することを特徴とする画像表示制御方法。

【請求項２２】　前記サーバー上に確保された前記画像描画メモリは、前記画像表示制御装置の前記画面メモリより大きいことを特徴とする請求項２１記載の画像表示制御方法。

【請求項２３】　前記サーバー上に確保された前記画像描画メモリの解像度は、前記画像表示制御装置の前記画面メモリの解像度より高いことを特徴とする請求項２１記載の画像表示制御方法。

【請求項２４】　前記画像表示制御装置の画面メモリに表示される画像データは、前記サーバー上に確保された前記画像描画メモリに描画される画像データの一部であることを特徴とする請求項２１記載の画像表示制御方法。

【請求項２５】　前記画像表示制御装置の画面メモリに表示される画像データをスクロールすると、前記サーバー上に確保された前記画像描画メモリに描画される画像データの一部が前記画面メモリに複写されることを特徴とする請求項２１記載の画像表示制御方法。

【請求項２６】　前記画像表示制御装置は、ホストコンピュータであることを特徴とする請求項２１記載の画像表示制御方法。

【請求項２７】　前記サーバーと前記画像表示制御装置は、ネットワークを介して接続されていることを特徴とする請求項２１記載の画像表示制御方法。

【請求項２８】　ホストコンピュータと接続された画像編集装置における画像編集方法であって、

前記ホストコンピュータから描画命令を入力する入力工程と、

前記入力工程で入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画する描画工程と、

前記描画工程で描画した前記画像データを前記ホストコンピュータの画面メモリに表示する為に、前記画像データを前記ホストコンピュータへ出力する出力工程とを有することを特徴とする画像編集方法。

【請求項２９】　前記画像編集装置に確保された前記画像描画メモリは、前記ホストコンピュータの前記画面メモリより大きいことを特徴とする請求項２８記載の画像編集方法。

【請求項３０】　前記画像編集装置に確保された前記画像描画メモリの解像度は、前記ホストコンピュータの前記画面メモリの解像度より高いことを特徴とする請求項２８記載の画像編集方法。

【請求項３１】　前記ホストコンピュータの画面メモリに表示される画像データは、前記画像編集装置に確保された前記画像描画メモリに描画される画像データの一部であることを特徴とする請求項２８記載の画像編集方法。

【請求項３２】　前記ホストコンピュータの画面メモリに表示される画像データをスクロールすると、前記画像編集装置に確保された前記画像描画メモリに描画される画像データの一部が前記画面メモリに複写されることを特徴とする請求項２８記載の画像編集方法。

【請求項３３】　前記画像編集装置は、サーバーであることを特徴とする請求項２８記載の画像編集方法。

【請求項３４】　前記ホストコンピュータと前記画像編集装置は、ネットワークを介して接続されていることを特徴とする請求項２８記載の画像編集方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、ネットワークに接続されたホストコンピュータ等の画像表示制御装置及びその方法とサーバー等の画像編集装置及びその方法と当該画像表示制御装置と当該画像編集装置から構成されるシステムに関する。

【０００２】

【従来の技術】従来、ネットワーク化の進展に伴い種々の端末が通信機能を持つようになると、相互接続性が非常に重要になってくる。例えば、Ｆａｘの場合、公衆網にアクセスする方法に関してはＣＣＩＴＴが規定したファクシミリ通信手順が存在するが、Ｆａｘサーバーとしてネットワーク化しようとする場合の手順は標準化されていない。

【０００３】また、リレーショナルデータベース（ＲＤＢ）に関しては、一応ＡＮＳＩのＳＱＬ言語が標準としての地位を確立しているが、ＲＤＢへのアクセスプロトコルやフロントエンドのインタフェース（ＡＰＩ）は各社バラバラである。印刷に関しては、ＯＳ毎の業界標準は存在するが、マルチベンダー環境におけるネットワークプリンタのアーキテクチャは標準化されていない。

【０００４】ＯＣＲに関しては、従来ネットワーク化の概念がなかったために、ネットワークＯＣＲのアクセス方法はこれから決まってくる。ネットワーク上の画像処理アクセラレータの概念も従来はなかった。このよう

に、データベースならデータベース、ネットワークプリンタならネットワークプリンタといった個別アプリケーション分野毎の標準化の努力はなされているが、分野をまたがる標準化はない。例えば、Ｆａｘサーバーで受信した画像をＯＣＲサーバーでコードに変換し、それをデータベースに蓄積するといった一連の作業を行うには、各サーバー間の連携はないから、クライアント側でＦａｘとＯＣＲとデータベースの３本のクライアント側プログラムを起動し順次作業を進めていくことになる。それでも、もし３本のプログラムが、最新のＯＳの提供する機能をフルに用いて、アプリケーション間通信を行えば、上記の作業は自動化できるかも知れない。しかし、最新のＯＳはアプリケーション間通信の仕組は提供するが、そのやり方はベンダー任せであり、何も決まっていない。

【０００５】

【発明が解決しようとする課題】しかしながら、従来のサーバーシステムにおいては、例えば、Ｆａｘサーバーで受信した画像をＯＣＲサーバーでコードに変換し、それをデータベースに蓄積するといった一連の作業を行うためには、各サーバー間の連携はないから、クライアント側でＦａｘフロントエンドとＯＣＲフロントエンドとデータベースフロントエンドの３本のフロントエンドプログラムを起動し順次作業を進めていくことになる。しかし、その作業は多くの「カット＆ペースト」や「一時ファイル」へのアクセスの繰り返しとなり苦痛を伴う。

【０００６】また、個々のサーバーへのアクセス方法は、サーバーが発明された歴史上の必然によりバラバラで、サービスの分野にまたがるアクセスはほとんど不可能である。本発明は、ネットワーク上に分散している独立サーバーを仮想的に統合し、巨大な仮想サーバーを論理的に構築することにより、サーバー毎の互換性をなくし、１本のアプリケーションから簡単に多くのサーバー機能を利用可能ならしむるためのものである。

【０００７】また、従来、小さなパーソナルコンピュータではメモリの不足により、大画面、高精細画像の編集はできなかった。例えば、４００ｄｐｉ（ドット／インチ）でＡ４サイズのフルカラー画像は、４７メガバイトものメモリを必要とする。パーソナルコンピュータでこの様に大きな画像を扱える様にするのはコスト的にも、速度的にも、非現実的であり、信頼性の面からも問題であった。なぜならシングルチップアーキテクチャの制約から、パーソナルコンピュータのＣＰＵパワーは、それほど大きくなく、速度的に遅くなる。また、ハードディスクをＲＡＭの一部に取り込む方法だと、スワップが発生する度に絶望的な遅延が発生する。また、今日のパーソナルコンピュータのＯＳの場合、この様に大きなメモリを扱っていると、クラッシュしてしまう事が多い。

【０００８】かかる問題点を解決するために本発明の目的は、サーバに描画命令を出力し、出力した前記描画命令を解釈してサーバー上に確保された画像描画メモリ上に描画された画像データを取得し、取得した画像データを画面メモリに表示する画像表示制御装置及びその方法を提供することである。

【０００９】かかる問題点を解決するために本発明の目的は、ホストコンピュータから描画命令を入力し、入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画し、描画した画像データをホストコンピュータの画面メモリに表示する為に、画像データをホストコンピュータへ出力する画像編集装置及びその方法を提供することである。

【００１０】かかる問題点を解決するために本発明の目的は、サーバに描画命令を出力し、出力した前記描画命令を解釈して、サーバー上に確保された画像描画メモリ上に描画された画像データを取得し、取得した前記画像データを画面メモリに表示する表示制御手段とを有するホストコンピュータと、ホストコンピュータから描画命令を入力し、入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画し、描画した前記画像データを前記ホストコンピュータの画面メモリに表示する為に、画像データをホストコンピュータへ転送するサーバーとから構成されるシステムを提供することである。

【００１１】

【課題を解決するための手段】上記問題点を解決するために本発明の画像表示制御装置は、画像処理を実行するサーバーと接続された画像表示制御装置であって、前記サーバに描画命令を出力する出力手段と、前記出力手段で出力した前記描画命令を解釈して、前記サーバー上に確保された画像描画メモリ上に描画された画像データを取得する取得手段と、前記取得手段で取得した前記画像データを画面メモリに表示する表示制御手段とを有する。

【００１２】上記問題点を解決するために本発明の画像編集装置は、ホストコンピュータと接続された画像編集装置であって、前記ホストコンピュータから描画命令を入力する入力手段と、前記入力手段で入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画する描画手段と、前記描画手段で描画した前記画像データを前記ホストコンピュータの画面メモリに表示する為に、前記画像データを前記ホストコンピュータへ出力する出力手段とを有する。

【００１３】上記問題点を解決するために本発明のシステムは、画像処理を実行するサーバーとホストコンピュータとがネットワークに接続されたシステムであって、前記ホストコンピュータは、前記サーバに描画命令を出力する出力手段と、前記出力手段で出力した前記描画命令を解釈して、前記サーバー上に確保された画像描画メモリ上に描画された画像データを取得する取得手段と、前記取得手段で取得した前記画像データを画面メモリに

表示する表示制御手段とを有し、前記サーバーは、前記ホストコンピュータから描画命令を入力する入力手段と、前記入力手段で入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画する描画手段と、前記描画手段で描画した前記画像データを前記ホストコンピュータの画面メモリに表示する為に、前記画像データを前記ホストコンピュータへ転送する転送手段とを有する。

【００１４】上記問題点を解決するために本発明の画像表示制御方法は、画像処理を実行するサーバーと接続された画像表示制御装置における画像表示制御方法であって、前記サーバに描画命令を出力する出力工程と、前記出力工程で出力した前記描画命令を解釈して、前記サーバー上に確保された画像描画メモリ上に描画された画像データを取得する取得工程と、前記取得工程で取得した前記画像データを画面メモリに表示する表示制御工程とを有する。

【００１５】上記問題点を解決するために本発明の画像編集方法は、ホストコンピュータと接続された画像編集装置における画像編集方法であって、前記ホストコンピュータから描画命令を入力する入力工程と、前記入力工程で入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画する描画工程と、前記描画工程で描画した前記画像データを前記ホストコンピュータの画面メモリに表示する為に、前記画像データを前記ホストコンピュータへ出力する出力工程とを有する。

【００１６】

【作用】以上の構成によれば、サーバに描画命令を出力し、出力した前記描画命令を解釈してサーバー上に確保された画像描画メモリ上に描画された画像データを取得し、取得した画像データを画面メモリに表示することで、クライアントでの画像描画処理をサーバーに代行させることができる。

【００１７】以上の構成によれば、ホストコンピュータから描画命令を入力し、入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画し、描画した画像データをホストコンピュータの画面メモリに表示する為に、画像データをホストコンピュータへ出力することで、クライアントでの画像描画処理を代行し画像編集処理できる。

【００１８】

【実施例】図１は本発明の一実施例を示すマルチメディアサーバーのシステム構成を説明するブロック図である。特に、センタサーバー１１を介した既存サーバー１０のアクセスを行う場合に対応する。

【００１９】なお、本実施例に示すマルチメディアサーバーにおいて、マルチメディア情報を扱うのに適したサーバーは、「統合化サーバー」と「ディジタル交換機」とから構成され、ネットワーク上に仮想的な１個の統合化サーバーを構築して各種のサービスを提供する。

【００２０】本実施例において、「統合化サーバー」とは、ネットワーク上に分散するサーバー群を論理的に統合したもので、従来型の種々のサービスを提供する複数の機能サーバーと、それらをコントロールする１個のセンタサーバーとからなる。センタサーバーは専用サーバーであってもよいし、クライアントが兼用するものであってもよい。

【００２１】クライアント（ホストコンピュータ）からセンタサーバーを見ると、ネットワーク上に分散する複数の機能サーバー群が、統合化した１個の巨大なサーバーのように見える。勿論個々のサーバーにも従来通り個別にアクセスしてもよいが、センタサーバーに「プロセススクリプト」を渡して以後の処理はセンタサーバーに処理を任せた方が、クライアントは次の作用に素早く移れるので効率的である。

【００２２】図において、１はサーバーをアクセスする「フロントエンド」としてのアプリケーションである。ここで、「フロントエンド」を簡単に説明する。

【００２３】もし、サーバーがクライアントと同じコンピュータに同居していたら、アプリケーションはサーバーの機能を直接呼び出して利用できる。しかし、サーバーがネットワークを介して、別のコンピュータ上にある時は、簡単ではない。なぜなら、アプリケーションは、通信プログラムを介してサーバーにアクセスしなければならないからである。

【００２４】これではアプリケーションのプログラマーは簡単にはサーバーにアクセスするプログラムが書けない。そこで、サーバーのベンダーは、たとえそのサーバーがネットワーク上にある場合でも、ローカルにある場合と同じ呼び方でアクセスする仕組みを用意するのが普通である。

【００２５】その仕組みは、サーバーの一部の様なもので、通信機能を持ち、クライアントにインストールされる。この小さなプログラムの事を、サーバーに対するフロントエンドと呼び、プログラマーは通信の事を何等意識する事なしにネットワーク上のサーバーにアクセスするアプリケーションを作る事ができる。

【００２６】ユーザがデータを入出力するのは、すべてこのアプリケーション１を通じて行う。２、４はクライアント／サーバー型の通信プログラム、３はローカルエリアネットワーク（ＬＡＮ）である。５は前記アプリケーション１に対して統合化された標準のアクセス環境を提供するセンタサーバーである。アプリケーション１からはこのセンタサーバー５が全てのサービスを提供するように機能する。センタサーバー５は、既存の各種サーバーを論理的に統合するのが基本機能なので、通常はサーバー本来のサービス機能は備えていない。そこで、サーバーマネジャー６を通じて既存サーバー１０にアクセスし、アプリケーション１からの要求に応える。７はクライアント通信プログラム、８はローカルエリアネット

ワーク（ＬＡＮ）である。９はサーバー通信プログラムであるが、扱うデータとインタフェースの作りは通信プログラム２、４と同じではない。

【００２７】既存サーバー１０をアクセスする言語（コマンド）は、それぞれのサーバーの構成との関係でサーバー毎の固有の言語を持っている。従って、アクセスするためのクライアント／サーバー通信プログラム７、９も独自のものである場合も多く、複数のサーバーを統合するためのセンタサーバー５と接続するクライアント／サーバー通信プログラム２、４とクライアント／サーバー通信プログラム７、９とは異なる場合もある。

【００２８】また、ＬＡＮ８に関しては同様の理由によりＬＡＮ３と同一である必要はない。例えばＬＡＮ３がＡｐｐｌｅＴａｌｋ（商品名）でＬＡＮ８がＥｔｈｅｒｎｅｔ（商品名）であってもよい。既存サーバー１０をコントロールするサーバーマネジャー６は、既存サーバー１０のフロントエンドとして動作する訳であるが、クライアント通信プログラム７とのインタフェースは、やはり既存のＡＰＩ（Ａｐｐｌｉｃａｔｉｏｎ　Ｐｒｏｇｒａｍｍｉｎｇ　Ｉｎｔｅｒｆａｃｅ）となる。

【００２９】一方、センタサーバー５は全てのサーバーに共通のアクセス方法を提供し、サーバー／クライアント通信プログラム２、４を介して提供するＡＰＩも共通のものとなる。

【００３０】従って、センタサーバー５とサーバーマネジャー６との間でアーキテクチャの違いを吸収するためのプロトコル変換とデータ変換を行うことになる。

【００３１】このため、センタサーバー５とサーバーマネジャー６は同一コンピュータの中にあって密接な関連の下で動作する。そして、センタサーバー５とサーバーマネジャー６との機能を統合した手段がセンタサーバーとしての機能である。従って、センタサーバー５は、機能を明確に区別する必要のある時はサーバーーサーバーと呼ぶ。

【００３２】図２は、図１に示したセンタサーバー５を介した既存サーバーのアクセス方法を説明する図である。

【００３３】図において、クライアント２０～２５は、共通のアクセス方法によりセンタサーバー２６にアクセスすると、センタサーバーはクライアントからの要求に基づき、必要な機能サーバー２７～３０に個別のアクセス方法を通じてアクセスする。従って、各クライアント２０～２５は個別の機能サーバーＡ～Ｄへのアクセス方法はおろか、それらの存在すら関知する必要はない。

【００３４】この様に構成されたマルチメディアサーバーにおいては、ネットワークを介して転送するセンタサーバーが各クライアントからの発行される所定のプロセススクリプトとデータとが一対となるメッセージを受信し、該受信したメッセージを解釈して、各機能サーバーに対する通信プロトコル変換及びデータ変換を施して各機能サーバーに対する連続した処理を代行しながらそれぞれのプロセススクリプトとデータを前記ネットワークを介して転送するので、クライアントの各プログラムはセンタサーバーを複数の機能を実行させる場合であっても、センタサーバーにその複合化された機能処理に対応するプロセススクリプト及びデータを転送するだけで、所望の結果データを得ることが可能となる。

【００３５】ここで「プロセススクリプト」を説明する。プロセススクリプトとは、通信プロトコルによってデータ通信をする基本単位で、コマンドがテキストで記述されている。コマンドとデータを一緒に送る事もできる。通常は機械と機械の間の通信を行う為に用いるので、コマンドはテキストであってもバイナリであっても差はない。しかし、テキストは人間が読めるので、機械が電子メールを介して人間に送り、その結果を電子メールで送ってもらうと言った変わった応用も可能である。

【００３６】プロセススクリプトの最大の特徴は、通信プロトコル自身もスクリプトで送れるので、通信手順をダイナミックに変更しながら通信を成立させる事が可能となる。例えば、Ｇ４Ｆａｘの通信手順を思い出してみよう。サービスが、極めて厳格なプロトコルの上に構築されてるので、Ｆａｘを受け取ったら同じ文書を何箇所かに配信するなどという芸当は、標準プロトコルの中ではできない。

【００３７】プロセススクリプトは、通信手順というより、サービスの手順を記述したものだから、通信方法や通信媒体はなんでも良く、基本サービスを理解できる端末なら、複合動作も簡単に指定できる。一例として、日本からアメリカのオフィスにあるプリンターにＤＴＰ文書を印刷する事を考えて見よう。普通はネットワークが繋がっていないので、この様な事はできない。新しい方式ならＤＴＰアプリが吐き出す印刷コマンドをプロセススクリプトで包んで、電子メールで相手に届ける。相手がメールサーバーなら自動的に、そうでなければ手動で、受信したスクリプトをスクリプトマネジャに送ると、その内容が解析され、相手のオフィスのプリンターに印刷出力が出て来る。

【００３８】また、各クライアントからの発行される所定のプロセススクリプトとデータとが一対となるメッセージを受信し、該受信したメッセージを解釈して、各機能サーバーに対する通信プロトコル変換を施して各機能サーバーに対するプロセススクリプトを前記ネットワークを介して転送するセンタサーバーを前記ネットワークに接続したので、クライアントの各プログラムはセンタサーバーを複数の機能を実行させる場合であっても、センタサーバーにその複合化された機能処理に対応するプロセススクリプトを転送するだけで、所望の結果データを得ることが可能となる。

【００３９】さらに、センタサーバーからプロセススクリプトを受信した各機能サーバーは、プロセススクリプ

ト中のＩＤ情報に基づいてネットワークを介してクライアントからデータを受信するので、クライアント間と各機能サーバーにスクリプトとデータとを独立して転送することが可能となる。

【００４０】また、ネットワークに接続されたセンタサーバーが各クライアントからの発行される所定のプロセススクリプトとデータとが一対となるメッセージを受信し、該受信したメッセージを解釈して、各機能サーバーに対する通信プロトコル変換及びデータ変換を施して各機能サーバーに対するプロセススクリプトとデータを前記ネットワークを介して転送し、交換機が各クライアントとセンタサーバーとの電話回線を交換するので各機能処理実行中に音声情報等のリアルタイム情報を並行してクライアント相互間でマルチセッションしながら複合情報を転送することが可能となる。

【００４１】これにより、あらゆるサーバーのサービスを１種類の統一のとれた簡単なアクセスで処理手順を標準化し、シームレスで無駄の無いサーバー環境が提供でき、例えばＦＡＸサーバーで受信した画像をＯＣＲサーバーでコードに変換し、さらに、ファイルサーバーでデータベースに蓄積するといった一連の複合作業を、１つのスクリプトで連続して処理することが可能となる。従って、従来同様のクライアント側で処理に必要とされるような、クライアント側でＦＡＸフロントエンドとＯＣＲフロントエンドとデータベースフロントエンドの計３本のフロントエンドプログラムを起動し、順次作業を進める、カット＆ペーストや一時ファイルへのアクセスの繰返し処理を大幅に軽減することができる。

【００４２】また、ネットワーク上に分散している独立サーバーを仮想的に統合できるため、サーバー毎の互換性をクライアントが意識することなく、１本のアプリケーションから簡単に、かつ多くのサーバー機能もしくは複合的なサーバー機能を利用することが可能となる。

【００４３】さらに、従来、個別の企業／業界において、ばらばらに開発されてきたサーバー装置へのアクセス方法を統合化し、ネットワーク上の既存のばらばらなインタフェースを有するサーバーに１つの統合されたアクセス方法でアクセスする手段を提供することが可能となる。以下、それぞれの実施例に分けて詳述する。

【００４４】図３は本発明に係るマルチメディアサーバーにおけるメッセージの構造を説明する図である。

【００４５】図３に示すように、本実施例におけるメッセージは、プロセススクリプトフォーク（プロセススクリプト）とデータフォーク（データ）とから構成されている。また、プロセススクリプトとデータは、それぞれ共通の内容を持つタグエレメント（タグ）を備えている。

【００４６】このタグには、メッセージが作成された「時間」、メッセージが消去されるべき「ライフタイム」、一連の「ＩＤ番号」、データの「種類」、データを作成したアプリケーションの「サイン」等が記載される。

【００４７】この内「ＩＤ番号」は必須であるが、他はオプションである。

【００４８】図４は本発明に係るマルチメディアサーバーにおける第１のプロセススクリプトの送出手順を説明するブロック図である。

【００４９】通常、クライアント４１、センタサーバー４３、機能サーバー４４は、全て同一のＬＡＮ上に存在する場合が多い。そして、機能サーバー４４が本発明に準拠したプロセススクリプトを理解できるサーバーの場合には、センタサーバー４３がクライアント４１にその事を指示し、クライアント４１はメッセージ４２をプロセススクリプト４５とデータフォーク（データ）４６とに切り放して、それぞれセンタサーバー４３と機能サーバー４４に別々に送信することができる。センタサーバー４３は、受信したプロセススクリプト４５に必要な編集を施した後、機能サーバー４４に新たなプロセススクリプト４７を送り届ける。機能サーバー４４にとっては、クライアント４１もセンタサーバー４３も共にクライアントであるから、別々に届いたプロセススクリプト４７とデータ４６を結合し、データに必要な処理を施して結果のメッセージ４８をＬＡＮを介してクライアント４１に返す。

【００５０】なお、本実施例において、プロセススクリプトとは、センタサーバー４３が機能サーバー４４群を使って一連の仕事をするための手順を記述したプログラムリストであり、センタサーバー４３はそれに基づいて一連の作業を実行し、最終結果のメッセージ４８だけをクライアント４１に返すので、クライアント４１の負荷は、従来型の機能サーバー群に個別にアクセスを繰り返していくよりも極めて小さくなる。

【００５１】プロセススクリプトは、クライアント４１のデバイスドライバーが自動生成することもあり、該プロセススクリプトはクライアント４１にとっては、通信プロトコルの一連の関数コールの集りにしか過ぎず、センタサーバー４３にとっては一連の通信プロトコルの中から生成されるスクリプト言語であり、機能サーバー４４群にとっては通信プロトコルそのものである。

【００５２】プロセススクリプトの特徴として、従来のコンピュータ通信の形態を大きく分けると、

リアルタイム処理

バッチ処理

の２つがあった。

【００５３】バッチ処理は、大型コンピュータ（メインフレーム）で処理する時に良く使われた手法で、大勢のユーザ（クライアント）が処理を集中させた場合、ジョブをキューに順番に溜め込み、順番に処理してユーザに返すというやり方である。結果がいつ出てくるのか分からないという欠点があり、どうしても進捗を見たい場合

には、キューを覗いたりしなければならなかった。

【００５４】リアルタイム処理は、プロトコルのやり取りによって何時でも進捗が分かる仕組みが用意されており、そのプロセスの中で必要な処理が実行される。

【００５５】一見便利に見えるリアルタイム処理であるが、これにはネットワークもリアルタイム系でなければならないという条件が付く。例えば、ネットワークがインターネットの様な「ストアード・フォーワード」型の場合、ハブを何箇所か中継して行くと、その度に数分の遅延が生ずる。この様なネットワークでは、リアルタイム処理が使えない。

【００５６】プロセススクリプトは、この様な「ストアード・フォーワード」型のネットワークに最適な通信方法で、リアルタイム処理とバッチ処理の中間的な特性を持っている。スクリプトとは「記述する」という意味で、コマンドがテキストで記述されており、その分プロトコルが簡単になる。

【００５７】リアルタイム処理の例を、図５に示す。この図では、ユーザが「データを送るよ」というコマンドを発行し、サーバーから「いいよ」という返事をもらったら、「データ」を送り、「完了」という返事に対し、「了解」という送達確認を送っている。この様なプロトコルが使えるのは、間に何もハブを介さないユーザとサーバーのエンドツーエンドの通信だからである。

【００５８】これがもし図６の様に、ハブ１、２、３を介在する通信の場合どうなるかと言うと、同図の様に「データを送るよ」から「いいよ」まで順番に遅延が蓄積して行き、「データ」送信にたどり着くだけでも大変で、そのうちどこかの通信接続が（タイムアウト等で）切れてしまう。全く実用にならない。

【００５９】図７はストアードフォーワード型のネットワークを考慮したバッチ処理の例である。「データを送るよ」に対して「いいよ」の返事は、近接したハブからもらっている。この様にすれば、比較的早く「データ」を送れる。しかし、この方法でも、途中で何らかの通信エラーが発生した時の回復ルーチンは、気が遠くなるほど複雑である。「データを送るよ」が途中で紛失した場合ならなんとかなるだろうが、「いいよ」が紛失したら、その回復は多分不可能であろう。

【００６０】図８はプロセススクリプトを用いた通信方法の例である。「処理して返事しろ」というコマンドと「データ」がセットになって順に送られる。処理結果も「処理結果だよ」というコマンドと処理済みの「データ」がセットで送られるので、このネットワークではこれより早い通信方法は無い。通信異常でプロセススクリプトが紛失した場合でも、コマンドとデータがセットになっているので回復も容易である。

【００６１】図９は本発明に係るマルチメディアサーバーにおける第２のプロセススクリプトの送出手順を説明するブロック図であり、特に、機能サーバーが既存のサーバーである場合のアクセス方法を示したものであり、図４と同一のものには同一の符号を付してある。

【００６２】この図において、メッセージ４２のプロセススクリプトとデータは分離せずにセンタサーバー４３にリクエストとして送られる。リクエストされた機能がセンタサーバー４３に無い場合には、その機能を代行してくれる機能サーバー４４を捜し、それに既存のアクセス方法でクライアント４１のリクエストを伝え結果を受け取る。その結果はメッセージ４８としてクライアント４１に返される。〈通信プロトコル〉マルチメディアサーバーにおけるセンタサーバーの役割の一つは、種々のサーバーのバラバラなアクセス方法を標準化することにある。アプリケーションは、標準化されたインタフェースを持つセンタサーバーにアクセスすれば、そこから個々のサーバーへのアクセスはセンタサーバーが代行してくれる。標準化する項目は、大きく分けると、データの標準化とアクセス方法の標準化の２つある。

【００６３】データの標準化は一本化ではなく、複数化である。特定のアプリケーションに依存しない、業界標準のフォーマットで標準化し、かつ、標準化されないデータのやり取りも許容する。

【００６４】主なフォーマットとして現在使用されているものは、ＱｕｉｃｋＤｒａｗ、ＧＤＩ、ＲＴＦ、ＴＩＦＦ、ＰＩＣＴ、Ｂｉｔｍａｐ、ＰｏｓｔＳｃｒｉｐｔ、ＥＰＳ、Ｇ３／Ｇ４、ＰＣＬ、ＨＰ－ＧＬ、ＡＮＳＩＳＱＬ、ＡＳＣＩＩＴｅｘｔ、ＵＮＩＣＯＤＥＴｅｘｔ、Ｂｉｎａｒｙ　Ｄａｔａ等（登録商標及び商品名を含む）である。

【００６５】一方、アクセス方法の標準化は一本化に近い。統一されたアクセス方法により、さまざまなサービスを統一的に簡単に利用できる。

【００６６】ただし、業界標準のアクセス方法はサポートしない訳にはいかない。一方、図３で説明したように、伝送する内容はメッセージ、すなわち、「プロセススクリプト」と「データ」とがセットになっている。「データ」は、処理されるべき情報そのものであり、「プロセススクリプト」はその情報をどう扱うかを記述したコマンドシーケンスである。「プロセススクリプト」と「データ」のセットがサーバー間を伝送され、必要な処理がなされていく。「プロセススクリプト」の受渡し方法を標準化することにより、さまざまなサーバーに同じアクセス方法でアクセスできる。

【００６７】また、プロトコルの異なるサーバーに対しては、センタサーバー４３がゲートウェイとして機能する。例えばクライアント４１からＦａｘサーバーにアクセスするための関数コールが図１０の（ａ）に示す内容が発せられ、その結果、図６の（ｂ）に示すようなスクリプトがセンタサーバー４３に渡される。

【００６８】図１０は図９で用いるプロセススクリプトの例である。（ａ）はプロセススクリプトを生成する時

の関数。（ｂ）は実際にネットワーク上に送出されるスクリプト部分である。この場合、Ｇ３ファクシミリを例に取っているので、データとしてはＭＭＲデータが流れるが、省略してある。

【００６９】ＭＭ＿ｏｐｅｎでサーバーのアドレス（ｓｅｒｖｅｒ）とサービス種別（ｆａｘ＿ｓｅｒｖｉｃｅ）を指定する。

【００７０】ＭＭ＿ｓｅｎｄでｆａｘ＿ｓｅｒｖｉｃｅの形式はＧ３である事を指定する。送信先Ｆａｘ番号（ｄｅｓｔｉｎａｔｉｏｎ）もセットする。

【００７１】ＭＭ＿ｄａｔａで送信データバッファのサイズ（ｌｅｎｇｔｈ）と中身（ｃｏｎｔｅｎｔ＿ｂｕｆｆｅｒ）を送る。

【００７２】ＭＭ＿ｃｌｏｓｅでこのサービスを終了させる。プロセススクリプトは柔軟なので、この時はリアルタイム系に近い使い方をしている。

【００７３】図１１では、図１０で簡略化して書いたシーケンスを詳細に説明している。

【００７４】アプリケーション、クライアントとセンタサーバーとの通信プロトコルは、図１１に示す手順に従うものとする。

【００７５】なお、アクセスシーケンスとして、以下の２種類の方式がある。

【００７６】第１は、ストアードフォーワード方式で、蓄積交換であり、プロセススクリプトとデータのペアを伝送し終わったら、処理結果の成功／失敗に関係なく通信を一旦終了し、処理の最終的な結果は、改めて通信を再開して調べる。サーバーに作業を委託し終わったら、クライアント側では次の作業に移れるので、資源の利用効率が高まる。

【００７７】第２は、リアルタイム方式であり、最終的な処理結果が確定するまでは通信の接続を確保し続ける方式で、クライアントの作業が拘束される欠点はあるものの、処理の信頼性は高い。

【００７８】以下、本実施例におけるディジタル交換機の機能処理について説明する。

【００７９】本実施例のディジタル交換機は、公衆回線交換網および構内回線交換網を接続制御する交換機で、センタサーバー４３からの接続指示に基づいて呼制御を行える。ＬＡＮを経由した遅延のある接続がなされている複数のクライアントの間に、構内回線交換網を経由した遅延のない第２の接続を行うことにより、リアルタイム性を要求される情報（音声）等を伝達できる。

【００８０】図１２は本発明におけるセンタサーバーと交換機を結合したマルチメディアサーバーとによるシステム構成を説明するブロック図であり、図２の応用例である。

【００８１】図１２において、６１～６５は各種の機能サーバー、６６～７０はクライアント、７１は各機能サーバー６１～６５を統合するセンタサーバー、７３はローカルエリアネットワーク（ＬＡＮ）、７２は内線／外線の電話回線を交換する交換機（ＰＢＸ）である。ＬＡＮ７３を介したデータ転送は、パケット化されて伝送されるため、音声や動画等のリアルタイム系の伝送には適さないが、全てのクライアントが常に接続されているという特徴を生かして、接続制御の不要なデータ転送や、同報通信に適する。

【００８２】一方、ＰＢＸ７２は音声や動画等のリアルタイム系の伝送に適する。ＬＡＮとＰＢＸの特徴を生かして、パソコン会議システム（ＰＣ会議）を構築することができる。いま、クライアント６６から６７にアクセスしてＰＣ会議を行う場合を考える。クライアント６６はノード７５を介してセンタサーバー７１に対し、クライアント６７とＰＣ会議を開始するための開始命令スクリプトを発行する。センタサーバー７１は、クライアント６７のマルチメディアクライアントデーモン（ＭＣＤ）と接続しＰＣ会議の開始を指示するスクリプトを発行する。ここで、デーモンとはバックグラウンドで常に走っているプログラムのことをいう。そのスクリプトにはＰＣ会議のＩＤ番号が書かれているので、以後そのＩＤ番号宛にクライアントからスクリプトを送り合うことにより、各クライアント間でデータ通信が可能となる。

【００８３】一方、同時に、センタサーバー７１は、制御線７４を介してＰＢＸ７２にアクセスし、クライアント６６と６７の近く（または内蔵）の電話機同士を接続し、音声の伝送を可能にする。この様にして各クライアント６６、６７のオペレータはコンピュータの画面上の共通の画面を見ながら音声で会話をし、ＰＣ会議を行う事ができる。ＰＣ会議は、１対１の接続である必要は無く、複数のクライアントを同時に接続する事もできる。

【００８４】これにより、パケット伝送を主体とし、リアルタイム系のデータ伝送を苦手とするＬＡＮと、音声／動画の交換を得意とするが、同報通信やインテリジェントな制御を苦手とする交換機を有機的に結合し、市販のパソコンに何等特別なオプションパーツを付加接続することなく、パソコン会議システムを構築することも可能となる。

【００８５】以下、高速データハイウェーを利用したサービス機能について説明する。

【００８６】なお、本実施例では、今日比較的入手し易いコンポーネントを想定してシステム構築を考えたが、ＦＤＤＩ（Ｆｉｂｅｒ　Ｄｉｓｔｒｉｂｕｔｅｄ　ＤａｔａＩｎｔｅｒｆａｃｅ：米国ＡＮＳＩ規格）の様な１００Ｍｂｐｓないしそれ以上の伝送レートを有するネットワークが利用できる場合には、データパケットと音声回線をマルチプレックスしても良い。

【００８７】同様に本実施例では、交換機ＰＢＸに接続する公衆網として、ＩＳＤＮを想定しているが、Ｂ－ＩＳＤＮであっても構わない。また、米国のＳＭＤＳクラス公衆網が利用できる場合にはＬＡＮ側の接続を外側に

接続しても構わない。

【００８８】また、多少速度を犠牲にしても構わない場合には、ＵＮＩＸのＳＬＩＰ（ＳｅｒｉａｌＬｉｎｅＩＰ）の様に、ＬＡＮ間接続をＩＳＤＮ網を介して接続しても構わない。また、現状の技術でも、ＩＳＤＮルータを用いればＩＳＤＮ網を介してＬＡＮ間接続が可能である。

【００８９】以下に、本発明が適応可能なサービスの例を示す。

【００９０】（分野）－（サービス）

印刷－カラー印刷

スキャン－カラースキャナ

ＯＣＲ－ＯＣＲ、ファイルキーパー、伝票処理

翻訳－日英

ファイルシステム－ＮＦＳ

データベース－テキスト、静止画、動画、音声

会議システム－共用ウインドウ＋リアルタイム音声

メール－テキスト、静止画、動画、音声

投稿システム－テキスト、静止画、動画、音声

画像処理－ＣＭＭ、フィルタリング、画像処理アクセラレータ

Ｆａｘ－Ｆａｘ送信／受信／配信

ＭＨＳ－テキスト、Ｆａｘ、ＣＡＴＳ

予約システム－会議室予約等である。

【００９１】以下、ＯＣＲサービスを例としてサービス機能処理について従来と本実施例とを対比して説明する。

【００９２】いま、１０枚の原稿をスキャンし、それをＯＣＲにかけて文字コードに変換し、それをテキストファイルとしてデータサーバーのディスクにセーブする作業を考える。

【００９３】従来では、（スキャナで原稿のスキャンしてクライアントに伝送）（イメージデータをＯＣＲサーバーに再転送ＯＣＲに掛けてテキストデータに変換）（テキストデータをクライアントに伝送）（テキストデータをデータサーバーのディスクに転送）を１０回繰り返す。

【００９４】一方、本実施例では、クライアント→センタサーバーにプロセススクリプトを伝送（スキャナで原稿のスキャン→ＯＣＲサーバーに伝送）（ＯＣＲに掛けてテキストデータに変換）（テキストデータ→データサーバーのディスクに転送）センタサーバー→クライアントに結果を知らせる。

【００９５】従来例では、クライアントが介在するデータ転送が４回必要で、通常のパソコンではハードディスク容量の制限から、１０枚もの原稿を一気にスキャンする事はしないので、最悪４０回のデータ転送が必要になる。それに対してクライアントが介在するデータ転送は２回で済み、その他はすべてサーバー同士の通信である。全体のデータ転送も減っている。クライアントが関係しないデータ通信／処理はカッコで示してある。データ転送の回数が減るという事は、コンピュータを操作する回数も減る事になるので、作業の大幅な自動化が実現される。

【００９６】以下、スキャンサービスを例としてサービス機能処理について従来と本実施例とを対比して説明する。

【００９７】Ａ３サイズの原稿をネットワークスキャナで４００ｄｐｉフルカラーでスキャンし、それに色処理して、プリンタサーバーに印刷する作業を考える。

【００９８】従来例では、原稿のスキャン→クライアントに伝送イメージデータ→色処理後のイメージデータ→プリンタサーバーに伝送印刷

一方、本実施例では、クライアント→センタサーバーにプロセススクリプトを伝送（原稿のスキャン）（イメージデータ→色処理）（色処理後のイメージデータ→プリンタサーバーに伝送）（印刷）センタサーバー→クライアントに結果を知らせる。

【００９９】通常のパソコンではメモリ容量の制限があるので、実は、従来例というのは机上の計算値でしかなく、実際には９６ＭＢｙｔｅもの巨大なメインメモリを管理できるＯＳは事実上存在しないに等しい。

【０１００】以下、メールサービスを例としてサービス機能処理について従来と本実施例とを対比して説明する。

【０１０１】動画ファイルをサーバーに送って動画のプレゼンテーションを実行する場合。

【０１０２】従来例では、

巨大な動画ファイル→サーバーに伝送

一方、本実施例では、クライアント→センタサーバーに動画スクリプトを伝送（センタサーバー－機能サーバー間通信）センタサーバー→クライアントに結果を知らせる。

【０１０３】動画ファイルは一般的に極めてサイズが大きくなる。数１０分程度の映画で１ギガバイトを越える事もある。

【０１０４】この様なサイズのデータを扱うためには、従来のクライアントのメモリやハードディスクを増設し、場合によってはＣＰＵ本体も高速な物に交換する必要があった。

【０１０５】しかし、本実施例では、サーバーが動画データを記憶してくれるのでクライアントのメモリやハードディスクを増設する必要は無く、ＣＰＵも遅い物で構わない。ネットワークを介してマルチメディアサーバー（センタサーバー）にプロセススクリプトを発行すれば、動画の再生はサーバーが代行してくれる。

【０１０６】以下、メールサービスを例として会議システム機能処理について従来と本実施例とを対比して説明する。

【０１０７】複数のクライアントを同時に接続し、同じ

ファイルをオープンしてどこからでも書き込める電子黒板を用意し、同時に交換機にアクセスしてそれぞれの内線電話同士を接続し、パソコン会議を行う。

【０１０８】従来例では、音声の入らない、テキストベースの会議システム程度一方、本実施例では、クライアント→センタサーバーに会議用プロセススクリプトを伝送（センタサーバー－機能サーバー間通信）（クライアントとクライアント間のパソコン会議）（クライアント→センタサーバーに終了プロセススクリプトを伝送）伝送センタサーバー→クライアントに結果を知らせる。

【０１０９】以下、図１３を参照しながら本発明に係るマルチメディアサーバーを画像描画アクセラレータとして機能させる場合について説明する。

【０１１０】図１３は本発明に係るマルチメディアサーバーを画像描画アクセラレータとして機能させる場合の概念を説明するブロック図である。特に、本実施例では、マルチメディアサーバーをアップルコンピュータ株式会社のコンピュータ（登録商標：Ｍａｃｉｎｔｏｓｈ）の画像描画アクセラレータとして用いた場合の例である。

【０１１１】例えば、Ｂ５で４００ｄｐｉフルカラー程度以上のビットマップ画像の編集を行う為には、メインメモリが数十メガバイトも必要になり、従来のパソコンでは不可能であった。最新のＯＳの提供する仮想メモリ機能を利用しても、処理速度の点からＡ４で４００ｄｐｉフルカラー程度が限度であろう。

【０１１２】一方、本実施例のマルチメディアサーバーをネットワーク上の仮想メモリとして利用する事により、Ａ３サイズ程度以上のフルカラー高解像度画像の編集を可能にする。ここで、仮想メモリを説明する。

【０１１３】極めて巨大なメモリを利用する方法。実メモリとは、通常は半導体メモリ（あるいは主記憶）の事を言い、要求された巨大なメモリサイズのごく一部しか存在しない。それに対して、仮想メモリとは、ハードディスクやネットワーク上のサーバーのメモリ（あるいは補助記憶）の事を言い、アクセス速度は遅いが極めて巨大な記憶空間を有する。

【０１１４】主記憶と補助記憶をうまく組み合わせて、アプリから見たら、全メモリ空間が一様でシームレスな空間であるかの様に見せる方法を仮想メモリと言う。

【０１１５】アプリが主記憶に無い部分をアクセスしに行くと、メモリコントローラはエラーを検知するので、それまであった主記憶の情報を補助記憶に退避させ、逆に補助記憶から必要な情報を主記憶に呼び出し、かつ、その主記憶のアドレスを要求されたメモリ空間にスワップさせる事により、アプリから見たらあたかも巨大なメモリが実際に存在するかの様に見える。

【０１１６】具体的には、アプリケーション１０１は、モニタに何かを表示させる時には、必ずＯＳの一部である描画管理プログラム（商品名ＱｕｉｃｋＤｒａｗ）１０２に描画命令を送り、画面メモリ１０９へのラスタライズを依頼する。アプリケーションが画像メモリを直接アクセスする事は無い。

【０１１７】一方、ビットマップ画像をモニタに表示させる場合には、通常アプリケーションの責任で描画用バッファメモリを確保し、そこにバックグラウンドで描画した後、ＱｕｉｃｋＤｒａｗ１０２にメモリ転送の依頼を出す。

【０１１８】従って、描画用バッファメモリはアプリケーションの管理下にあり、それを仮想メモリにして実メモリをハードディスク上に設定する事は従来から行われていた。

【０１１９】本実施例においては、実メモリをネットワーク上のマルチメディアサーバー上に設け、画面がアップデートされた時にローカルメモリに部分コピーを行う。

【０１２０】１０３はマルチメディアサーバーの機能をクライアント側でアプリケーションに引き渡す為のＡＰＩ（Ａｐｐｌｉｃａｔｉｏｎ　Ｐｒｏｇｒａｍｍｉｎｇ　Ｉｎｔｅｒｆａｃｅ）である描画コマンドハンドラである。アプリケーション１０１がこのＡＰＩに対してＱｕｉｃｋＤｒａｗに対するコールと同じパラメータを渡してネットワーク上のサーバーに確保したメモリに描画する。

【０１２１】アプリケーションは必ずしもマルチメディアサーバーを意識して設計されている訳ではないので、アプリケーション１０１かＱｕｉｃｋＤｒａｗ１０２への関数コールをフックして描画コマンドハンドラ１０３に強制的に制御を渡しても構わない。ＱｕｉｃｋＤｒａｗの関数コールは全て例外処理で行っており、その処理を分岐させる事はたやすい。この場合、画像メモリに対するドロー描画コマンドはＱｕｉｃｋＤｒａｗにスルーさせる。マルチメディアサーバーを意識して設計されたアプリケーションの方が描画効率が高い事は言うまでもない。

【０１２２】一方、メモリに対するビットマップ描画コマンドはクライアント１０４、サーバー１０５を介してＰＤＬラスタライザー１０６に送られる。ビットマップ描画コマンドはここでビットマップデータにラスタライズされ、マルチメディアサーバー内のリモート実メモリ１０８に描画される。リモート実メモリ１０８は１ページ分のビットマップデータを全て記憶できる容量がある。それに対してローカル仮想メモリ１０７の容量は限定されたものである。

【０１２３】ローカル仮想メモリ１０７は、仮想的にリモート実メモリ１０８と重なり合うものであるが、実際にメモリが割り当てられている部分は、現在編集中の限られた領域のみである。編集中の領域が変更になれば、新しい領域の画像データをリモート実メモリ１０８からコピーして来る。

【０１２４】これにより、通常ではメモリ容量等の制限から、Ａ３、４００ｄｐｉフルカラー画像の編集等、従来ならば不可能な画像編集処理を安価な端末からネットワーク上の資源を利用して処理することができる。

【０１２５】図１４に本発明による画像描画アクセラレータのブロック図を示す。点線から右がサーバー側、点線から左がクライアント側である。図１３と同じ機能要素には同じ番号を付与してある。

【０１２６】例えば、クライアントがＭａｃｉｎｔｏｓｈ（商品名）の場合を例に取って説明すると、普通の使い方ではアプリケーション１０１が画面に描画する場合、ｇｒａｆｐｏｒｔと呼ばれる画面メモリ管理システム１０９をコールし、描画ポートを確保する。ｇｒａｆｐｏｒｔが確保する画面バッファメモリはビットマップ系であり、その解像度は７２ｄｐｉ（ドット／インチ）である。

【０１２７】アプリケーション１０１はＱｕｉｃｋＤｒａｗ（商品名）と呼ばれる一連の画面描画関数をコールする事ができ、通常それはＴｏｏｌＢｏｘコール（ＯＳのシステムコール）の形でＯＳ１０２Ｕ制御が渡される。ＯＳは受け取った画面描画関数コールをビットマップにラスタライズしてｇｒａｆｐｏｒｔに描画する。

【０１２８】本発明では、ＱｕｉｃｋＤｒａｗのＴｏｏｌＢｏｘコールをフック（盗んで）して一旦ネットワーク描画アクセラレータ１１０（ｎｅｔ＿ｇｒａｆｐｏｒｔ）に制御を渡し、それが改めてＴｏｏｌＢｏｘコールを発生させている。ネットワーク描画アクセラレータ１１０は描画コマンドハンドラ１０３とクライアン通信プログラム１０４を包含した部分を説明している。

【０１２９】ここで、クライアン通信プログラム１０４は、単に通信手段を提供する通信プログラムであるが、クライアント側と言うと、図の点線から左すべてを差して呼ぶ。サーバー側とサーバーの呼び方についても同様で、通信プログラム部分だけを指す場合とサーバー側と言うと点線から右全体を指す場合とがある。

【０１３０】さて、ネットワーク描画アクセラレータ１１０は、アプリケーション１０１からのＱｕｉｃｋＤｒａｗのＴｏｏｌＢｏｘコールをプロセススクリプトに変換してＬＡＮを介してネットワーク描画アクセラレータサーバー（サーバー側）にも伝える。それを受信した画像処理サーバ１１１は、例えば、４００ｄｐｉ（ドット／インチ）の高精細の解像度で高精細描画システム１０８（ｓｅｒｖｅｒ＿ｇｒａｆｐｏｒｔ）にクライアントと同じ画像を描画する。ローカル画面メモリ（画面メモリ管理システム１０９が管理するバッファメモリ）への描画が完了すると同時に、サーバー側では高精細な画像の描画が完了しており、必要ならそのまま印刷出力もできるし、フィルムに焼く事もできる。

【０１３１】図１５にネットワーク描画アクセラレータ１１０のブロック図を示す。クライアント側に関しては全てソフトウエアで実現する事も可能であるが、処理を高速化させる為にアクセラレータボードによるハードウエアで実現する事もできるこのアクセラレータはボードの形でクライアントのコンピュータのバス（νバス）２０９に挿入する。バスＩ／Ｏ２０６はボード内部の内部バス２０５と外部バス２０９との接続とアービトレーションを行う為のバスＩ／Ｏである。内部バス上にはＣＰＵ２０１、ＲＯＭ２０２、ＲＡＭ２０３、ＬＡＮＩ／Ｆ２０４があり、マイクロコンピュータを形成している。

【０１３２】割込ハンドラ２０７は、クライアントコンピュータ本体の割り込み信号線２１０をボード上のＣＰＵ２０１に導入する為のものである。ＢＵＳ　ＩＮＴ信号２１１は、Ｉ／Ｏ２０８を介してマザーボードに出力され、クライアントコンピュータ本体のＣＰＵにバスの使用を要求するための制御線である。ＢＵＳ　ＡＣＫ信号２１２は、バスが解放されたかどうかを知らせる制御線である。

【０１３３】アプリケーションが描画コマンドを発行すると、何らかのＱｕｉｃｋＤｒａｗルーチンがコールされ、割り込み（ＩＮＴ）が発生するから、その信号は割り込み信号線２１０を介して検知する事ができる。その割り込み信号に基づき、ＣＰＵ２０１はＩ／Ｏ２０８をイネーブルにし、ＢＵＳＩＮＴ信号２１１をグラントしてマザーボードのＣＰＵにバス解放要求を出す。バスが実際に解放されてトライステート状態になったら、ＢＵＳＡＣＫ信号が帰されてＣＰＵ２０１はバス２０９が使える状態になった事を知る。そこでＢＵＳＩ／Ｏ２０６をアクセスして内部バス２０５と外部バス２０９を結合し、所定の外部メモリ（図示せず）に書かれている割り込みジャンプテーブルのポインターをフェッチして来る。ポインターの指し示すメモリの内容を読むと、割り込みの種類が判別できるから、ＱｕｉｃｋＤｒａｗルーチンに関する割り込みであった場合には、ＲＡＭ２０３上でＱｕｉｃｋＤｒａｗコマンドをプロセススクリプトに変換してそれをＬＡＮ　Ｉ／Ｆ２０４からサーバーに伝送する。

【０１３４】サーバー側で、全画面のリモート実メモリに描画された内容のミラーイメージが、ローカル仮想メモリ１０７に反映される。実際には、画面メモリ１０９と同じ大きさのローカル実メモリ１１２の部分のみがコピーされる。アドレスコンバーター１１３は、外部バス２０９から見た実メモリ１１２が、あたかも仮想メモリ１０７の一部であるかの様に見せかける為のアドレス変換を行う回路である。仮想メモリ１０７の原点（左上）の座標が、Ｘ、Ｙであり、実メモリ１１２の矩形領域の座標が、ｘ、ｙであるなら、アドレスコンバーター１１３は外部バス２０９からアクセスされるアドレスに対し、Ｘ－ｘ、Ｙ－ｙ、の演算を実行する。

【０１３５】マザーボードのＣＰＵ（図示せず）がＣＲＴに表示している部分、すなわち実メモリ１１２の内部

だけで演算している場合は、仮想メモリ上の実メモリの位置関係は変化しない。しかし、スクロール等で実メモリの外側をアクセスしようとすると、割り込み信号１１４を発生させ、ＣＰＵ２０１に割り込み信号を与える。ＣＰＵ２０１は割り込みの原因を調べ、ＬＡＮ２０４を介してサーバーに対して新しい座標位置のビットマップ画像を送信する事を要求すると共に、アドレスコンバーターをセットしなおし、実メモリの相対位置を変更する。割り込み信号１１４は、実際にはアドレスコンバーター１１３が生成する。

【０１３６】以下、図１６を参照しながら本発明に係るマルチメディアサーバーにおけるローカル仮想メモリ処理動作の概要について説明する。

【０１３７】図１６は発明に係るマルチメディアサーバーにおけるローカル仮想メモリ処理の概要を示すチャートである。例えばＭａｃＯＳによるＱｕｉｃｋＤｒａｗのアクセラレータ処理について説明する。

【０１３８】なお、本実施例のローカルメモリは、本来アプリケーションがＯＳに要求して実メモリとして確保する代りに、ネットワークで接続されたサーバー上に確保した実メモリを仮想的にリモートのメモリであるかのように処理する。従って、ローカル仮想メモリは、ローカルなクライアントマシン上に仮想的に存在し、実メモリはリモートのサーバー上に実在する。

【０１３９】具体的には、アプリケーションが描画を開始する時、ＭａｃＯＳに対してメモリ確保コマンドとして、例えばＮｅｗＰｔｒ（Ｎｅｗ　Ｐｏｉｎｔｅｒの略でメモリ領域を確保するコマンド）を発行する。描画コマンドハンドラは、そのコマンドをストール（盗んで）してプロセススクリプト（Ｐスクリプト）に変換し、クライアント／サーバーの通信路を介してＰＤＬスタライザ／サーバー（以後、リモート描画エンジン）にプロセススクリプト（Ｐスクリプト）を送る。

【０１４０】リモート描画エンジンは、リモートのＯＳ（図示しない）にｍａｌｌｏｃコマンド（ｍｅｍｏｒｙ　ａｌｌｏｃａｔｅの略でＮｅｗ　Ｐｔｒと同じ働きをする）を発行して実メモリを確保するとともに、プロセススクリプトをクライアントに返す。そのプロセススクリプトは、描画コマンドハンドラからアプリケーションに対し、メモリのポインタとして返されるので、アプリケーションからはあたかもローカルに実メモリがあるかのように見える。

【０１４１】次に、アプリケーションは、ローカル仮想メモリに何かを描画しようとして、ＭａｃＯＳに「描画コマンド」を発行する。その「描画コマンド」は同様に描画コマンドハンドラにストールされ、プロセススクリプトに変換され、「リモート描画エンジン」に送られてリモート実メモリ上に実際に描画される。これは、４００ｄｐｉの高解像度で行われる。描画コマンドハンドラは、クライアントのモニタ上にも描画する必要があるので、クライアントのＭａｃＯＳに対しても描画コマンドをエミュレートして伝送する。この場合は、７２ｄｐｉの低解像度で描画される。

【０１４２】一方、描画の結果はプロセススクリプトに変換され、同様にクライアントに戻される。このプロセスは、必要な回数だけ繰り返し実行される（図中の太線の流れに対応する）。

【０１４３】描画が終了して印刷をする場合、本実施例によらなければ、ローカルのクライアント上で４００ｄｐｉの高解像度でラスタライズし、その膨大なデータをプリンタに伝送することになるが、本実施例によれば、描画が終了した時点で高解像度のラスタライズは「リモートエンジン」上で完了しているので、膨大なデータを送ることなしに、直ちに印刷を開始できる。従って、伝送されるのは、「印刷コマンド」のみである。

【０１４４】なお、完成したビットマップの高解像度の画像データを２次記憶装置等（図示しない）にセーブする場合にも、「セーブコマンド」を発行するものの、それによって「リモート描画エンジン」から送られてきた「Ｄａｔａ」を直接２次記憶装置等にセーブすれば、ローカルに実メモリを備える必要はなくなる。

【０１４５】以下、図１３のシステム原理図と図１７のフローチャートの両方を参照しながら、画像描画アクセラレータの動作を詳細に説明する。

【０１４６】図１７は画像描画アクセラレータ１１０の動作フローチャートであり、中央から左半分がクライアント側、右側がサーバー側の動作を示す。図では、クライアント／サーバー間の一般的な接続動作（例えばＡｐｐｌｅＴａｌｋやＴＣＰ／ＩＰなどのコネクション）は完了しているものとして説明している。なお、図１６の左半分のフローチャートに示すプログラムは、クライアント側のネットワーク描画アクセラレータ１１０のＲＯＭ２０２或いはＲＡＭ２０３に格納されており、ＣＰＵ２０１により実行される。また、図１６の右半分のフローチャートに示すプログラムは、サーバー側の不図示のＲＯＭ或いはＲＡＭに格納されており、不図示のＣＰＵにより実行される。

【０１４７】クライアントのアプリケーションが描画作業を実行する前には、その結果を保存するメモリを確保しに行く。その動作はＯＳに対する要求動作であり、一般的にはシステムコールと呼ばれる。Ｍａｃｉｎｔｏｓｈの場合にはツールボックスコールという。描画コマンドハンドラ１０３を介してＣＰＵ２０１はその関数コールをフックして、サーバー側に実メモリを確保する様に要求する。それがステップＳ１２と２３のやり取りに示される。

【０１４８】同時に描画コマンドハンドラ１０３を介してＣＰＵ２０１は、ステップＳ１３に示す様に、ローカルマシン上にも、画面メモリ用の最低限のバッファメモリを確保する。このローカルマシンの画面は７２ｄｐｉ

（ドット／インチ）なので、バッファメモリのサイズは極めて小さい。

【０１４９】次に、リモート実メモリ１０８と同じサイズのローカル仮想メモリ１０７のポインターのみを確保し、ステップＳ２３で帰された実メモリへのポインターで、上書きする。さらに、ステップＳ１３で確保した画面メモリのポインターを、仮想メモリ１０７空間の適当な部分に割り当てる。これによって、巨大なリモート実メモリ１０８に描画された絵の一部分がクライアントの画面に表示される様になる。

【０１５０】その後、描画コマンドハンドラ１０３を介してＣＰＵ２０１はイベント待ち（ステップＳ１５）になり、もし何らかの描画イベントが発生すると（ステップＳ１６）その描画コマンドをサーバー側に送信すると共にローカルマシンの画面メモリにも同じ描画を実行して（ステップＳ１７）画面表示する。描画コマンドを受信したサーバー側はＰＤＬラスタライザ１０６を介して不図示のＣＰＵが実メモリ１０８に高精細で描画する（ステップＳ２４）。

【０１５１】先にも述べた様に、ローカルマシンの画面には巨大なリモート実メモリ１０８に描画された絵の一部分が表示されているので、イベントとしては描画コマンドだけではなく、画面を上下左右にスクロールさせるスクロールイベント待ちになる（ステップＳ１８）。この場合、ローカル仮想メモリ１０７の表現するメモリ空間の一部に存在する様に割り当てられているローカル画面メモリ１０９の矩形領域を表すパラメータが、スクロールさせる分だけ変更になる（ステップＳ１９）。

【０１５２】ローカル画面メモリ１０９をローカル仮想メモリ１０７空間内で移動させると、ローカル画面メモリに書かれている絵と、リモート実メモリ１０８に描画された同じ位置の絵とが食い違って来る。そこで描画コマンドハンドラ１０３を介してＣＰＵ２０１はサーバーに対して新しい矩形領域のビットマップ画像を送る様に要求する（ステップＳ２５）。

【０１５３】サーバーの不図示のＣＰＵは指定された領域のビットマップ画像情報をクライアント側に送り（ステップＳ２６）、クライアント側のＣＰＵ２０１はそれをローカル仮想メモリ１０７にコピーする。すなわち、結果としてその画像情報はローカル画面メモリ１０９上に上書きされ、表示画面は新しくなる。

【０１５４】その後はイベント待ちを繰り返す。

【０１５５】アプリケーション１０１がビットマップ画像を編集する場合、ＯＳの描画機能１０２を用いてローカル実メモリ（図示せず）上で画像編集し、その結果を画面メモリ１０９を介してモニター（図示せず）に表示する。しかし、本発明で用いるローカルメモリは、その一部しか実体として持たない仮想メモリ１０７である。これによって巨大なメモリを用意する必要がなくなった。この場合、図１３の×印の様に本来全ての描画コマンドがＯＳのラスタライザ１０２に渡されていた所をストールして描画コマンドハンドラ１０３に導き、必要に応じてラスタライザ１０２でローカル仮想メモリ１０７と画面メモリ１０９を間接的に制御する。

【０１５６】ストールされた描画コマンドは、クライアント通信プログラム１０４、サーバー通信プログラム１０５を介してサーバー側のラスタライザ１０６に伝達される。そして高精細なリモート実メモリ１０８にビットマップ画像が描画される。リモート実メモリ１０８上にラスタライザされた画像は、そのまま印刷できる。

【０１５７】ローカル仮想メモリ１０７は、画面メモリ１０９と同じサイズのメモリで、リモート実メモリ１０８の一部分に相当する。ネットワーク画像処理を行う場合、まず１０４と１０５の間で一般的なクライアント／サーバーコネクションを張ったあと、アプリケーションが要求する画像メモリを確保する。メモリ確保要求は描画コマンドハンドラ１０３がフェッチし、実際にはローカルマシン上には仮想メモリ１０７が確保され、本当のメモリの実体は、ステップＳ１２と２３のやり取りにより、サーバー上のリモート実メモリ１０８として確保される。仮想メモリ１０７及びリモート実メモリ１０８の仮想的な大きさは一致しており、扱う原稿サイズ毎に異なる。

【０１５８】画面メモリ用のバッファも同時にステップＳ１３でローカルマシン上に確保される。その後はアプリケーション１０１は描画入力イベント待ちの状態になり、すなわち描画コマンドハンドラ１０３も入力待ちになる。これはステップＳ１５に示される。もし操作者がマウス等を操作して、何らかの描画のコマンドを入力すると、それがもし描画イベントならば描画コマンドがローカルマシンのラスタライザ１０２とサーバーのラスタライザ１０６の両方に送られて、画面メモリ１０９とローカル仮想メモリ１０７及びリモート実メモリ１０８の３箇所にそれぞれＣＰＵ２０１、サーバーの不図示のＣＰＵによって描画される。このステップを１６、１７、２４に示す。

【０１５９】もし、先の描画イベントが画面のスクロールの場合には、ローカル仮想メモリ１０７における画面メモリ１０９の参照座標が変化するが、そこには画像が存在しないので、ステップＳ１９及び２５の様にサーバーに矩形領域の変更が知らされ、ステップＳ２０、２６の様にリモート実メモリ１０８の所定の位置からビットマップ情報がクライアントに転送される。その後サーバーもクライアントもそれぞれ次のイベント待ちに入る。

【０１６０】以上説明したように、本実施例によれば、アプリケーションからオペレーティングシステムに対して発行される描画関数コールを捉えて前記サーバーに前記描画関数コールに対応するスクリプトを発行し、該スクリプトを解釈して前記サーバー上に確保された画像描画メモリ上に画像を描画し、該描画された描画画像デー

タを前記ネットワークを介して前記画像メモリから前記オペレーティングシステムに取り込み、画面メモリに描画するので、クライアントで処理可能なデータ量を越える大容量の画像描画処理を高速に行うことができる。

【０１６１】また、本実施例によれば、アプリケーションからオペレーティングシステムに対して発行される描画関数コールを捉えて前記サーバーに前記描画関数コールに対応するスクリプトを発行し、該スクリプトを解釈して前記サーバー上に確保された画像描画メモリ上に画像を描画し、該描画された描画画像データを前記ネットワークを介して前記画像メモリからローカル仮想メモリを介して前記オペレーティングシステムに取り込み、画面メモリに描画するので、クライアントで処理可能なデータ量を越える大容量の画像描画、特に編集画面に対応した描画処理を高速に行うことができる。

【０１６２】また、本実施例によれば、各クライアントから画像描画開始を指示するスクリプトを前記サーバーに発行し、該発行されたスクリプトに基づいて画像描画プログラムを起動するとともに、画像描画メモリを確保し、該確保された画像メモリに対応するパラメータを前記クライアントに返信して、前記クライアントからネットワークを介して受信する各描画命令に基づいて前記画像メモリに所望の画像を描画するので、クライアントの画像処理能力を越える画像編集処理を行うことができる。

【０１６３】また、本実施例によれば、サーバーは、各クライアントから画像描画開始を指示するスクリプトを解析して、クライアントの出力デバイスに対応する解像度よりも高解像度の画像描画を保証する画像描画メモリを確保するので、クライアントの画像処理能力を越える高解像度の画像編集処理を高速に行うことができる。

【０１６４】

【発明の効果】以上説明した様に本発明によれば、サーバーに描画命令を出力し、出力した前記描画命令を解釈してサーバー上に確保された画像描画メモリ上に描画された画像データを取得し、取得した画像データを画面メモリに表示することで、クライアントでの画像描画処理をサーバーに代行させることができるという効果を奏する。

【０１６５】以上説明した様に本発明によれば、ホストコンピュータから描画命令を入力し、入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画し、描画した画像データをホストコンピュータの画面メモリに表示する為に、画像データをホストコンピュータへ出力することで、クライアントでの画像描画処理を代行し画像編集処理できるという効果を奏する。

【図面の簡単な説明】

【図１】本発明の一実施例を示すマルチメディアサーバーのシステム構成を説明するブロック図である。

【図２】図１に示したセンタサーバーを介した既存サーバーのアクセス方法を説明する図である。

【図３】本発明に係るマルチメディアサーバーにおけるメッセージの構造を説明する図である。

【図４】本発明に係るマルチメディアサーバーにおける第１のプロセススクリプトの送出手順を説明するブロック図である。

【図５】本発明に係るリアルタイム処理を説明する図である。

【図６】本発明に係る他のリアルタイム処理を説明する図である。

【図７】本発明に係るバッチ処理を説明する図である。

【図８】本発明に係るプロセススクリプトを用いた通信方法を説明する図である。

【図９】本発明に係るマルチメディアサーバーにおける第２のプロセススクリプトの送出手順を説明するブロック図である。

【図１０】本発明に係るクライアントからセンタサーバー転送されるスクリプト例を示す図である。

【図１１】本発明に係るクライアント、アプリケーションとセンタサーバーと通信プロトコルの一例を示す図である。

【図１２】本発明におけるセンタサーバーと交換機を結合したマルチメディアサーバーとによるシステム構成を説明するブロック図である。

【図１３】本発明に係るマルチメディアサーバーを画像描画アクセラレータとして機能させる場合の構成を説明するブロック図である。

【図１４】本発明に係る画像描画アクセラレータのシステムブロック図である。

【図１５】本発明に係るネットワーク描画アクセラレータの具体的なブロック図である。

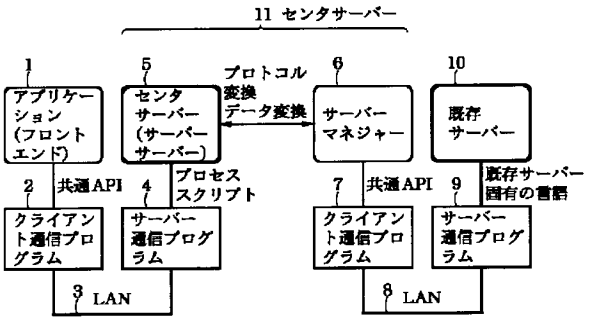
【図１６】本発明に係るマルチメディアサーバーにおけるローカル仮想メモリ処理の概要を示すチャートである。

【図１７】本発明に係る画像描画アクセラレータの動作を示すフローチャートである。

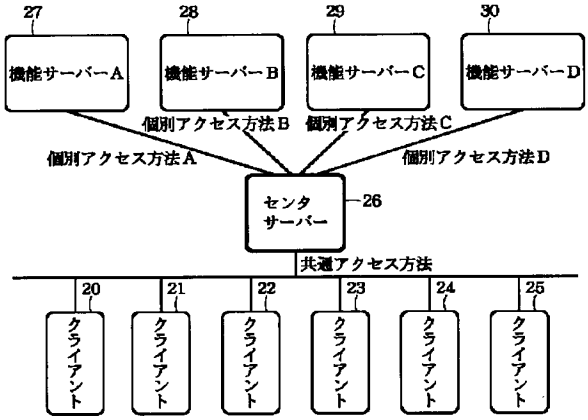
【符号の説明】

２０１　ＣＰＵ
２０２　ＲＯＭ
２０３　ＲＡＭ
２０４　ＬＡＮ／ＩＦ
２０５　内部バス
２０６　Ｂｕｓ　Ｉ／Ｏ
２０７　ＩＮＴ
２０８　Ｉ／Ｏ
２０９　外部バス

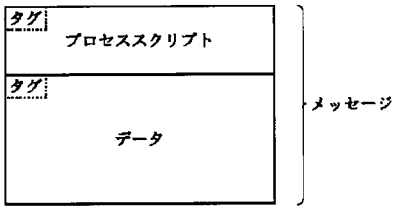
【図１】


11 センタサーバー
1
アプリケーション（フロントエンド）
5
センタサーバー（サーバーサーバー）
プロトコル変換
データ変換
6
サーバーマネジャー
10
既存サーバー
2
共通API
4
プロセススクリプト
7
共通API
9
既存サーバー固有の言語
クライアント通信プログラム
サーバー通信プログラム
クライアント通信プログラム
サーバー通信プログラム
3 LAN
8 LAN

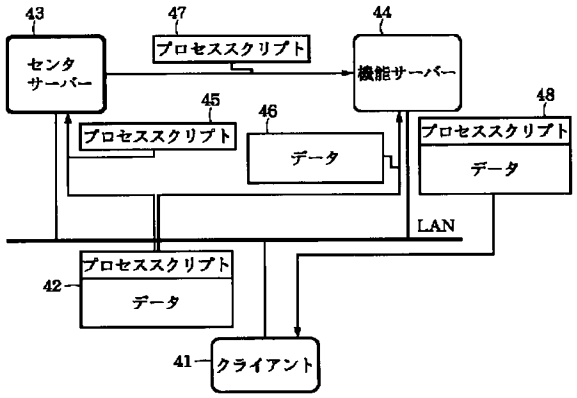
【図２】


27
28
29
30
機能サーバーA
機能サーバーB
機能サーバーC
機能サーバーD
個別アクセス方法A
個別アクセス方法B
個別アクセス方法C
個別アクセス方法D
センタサーバー
26
共通アクセス方法
20
21
22
23
24
25
クライアント
クライアント
クライアント
クライアント
クライアント
クライアント

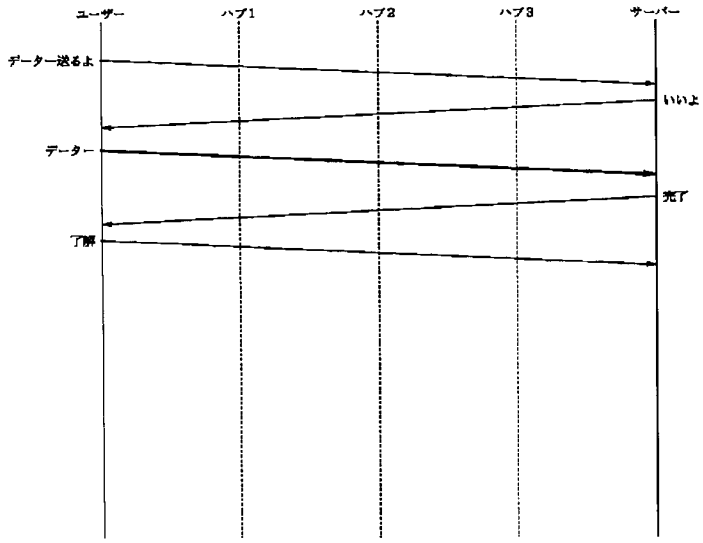
【図３】


タグ
プロセススクリプト
タグ
データ
メッセージ

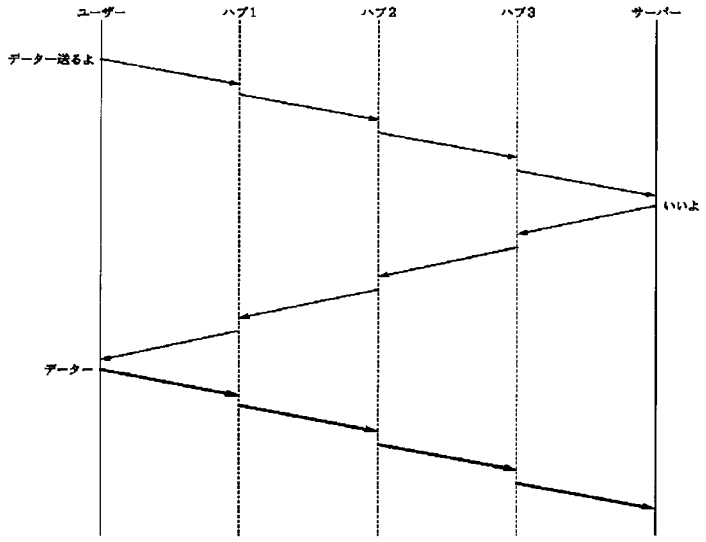
【図４】


43
センタサーバー
47
プロセススクリプト
44
機能サーバー
45
プロセススクリプト
46
データ
48
プロセススクリプト
データ
LAN
プロセススクリプト
42
データ
41
クライアント

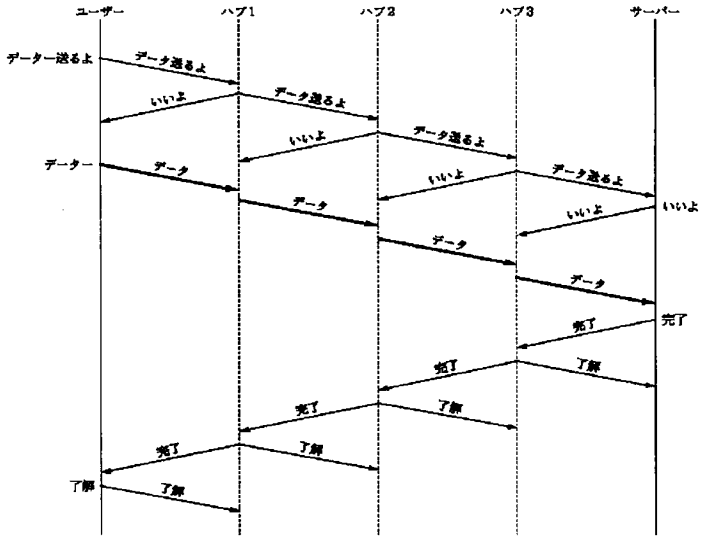
【図５】


ユーザー
ハブ1
ハブ2
ハブ3
サーバー
データー送るよ
いいよ
データー
完了
了解

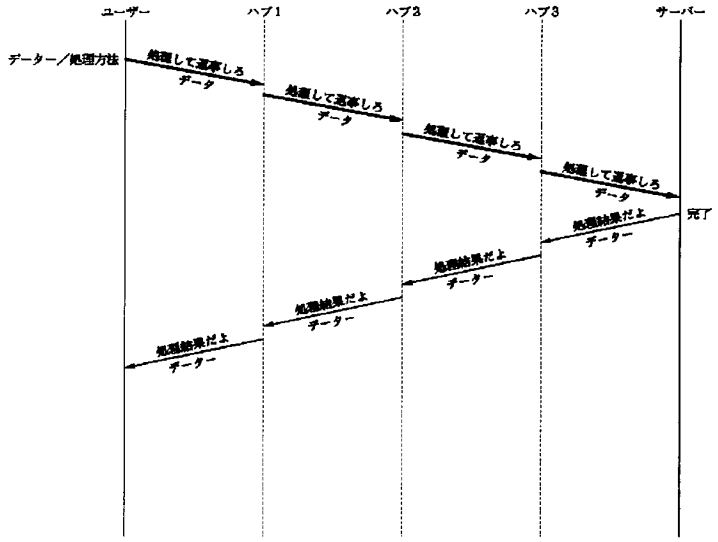
【図６】


ユーザー
ハブ1
ハブ2
ハブ3
サーバー
データー送るよ
いいよ
データー

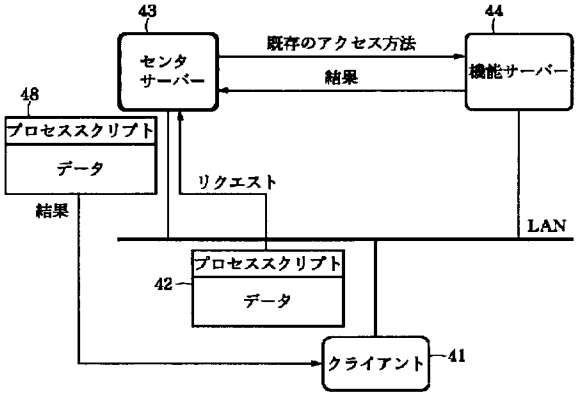
【図７】


ユーザー
ハブ1
ハブ2
ハブ3
サーバー
データー送るよ
いいよ
データー
データ
完了
了解

【図８】



【図９】



【図１０】

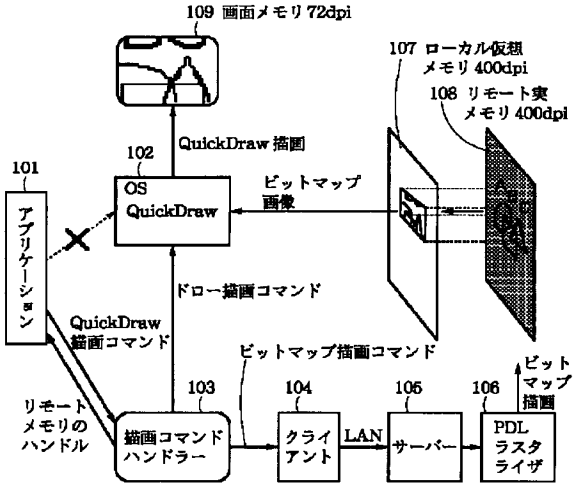
(a)

```
MM_open ("server","fax_service") ;
MM_send (service_id,"MM_send","G3","data","destination") ;
MM_data (service_id,"MM_send","data_id","length","content_buffer") ;
MM_close (service_id) ;
```

(b)

```
"MM_open server fax_service"
"service_id MM_send G3 data destination"
"service_id MM_send data id length content"
"service id MM_close"
```

【図１３】

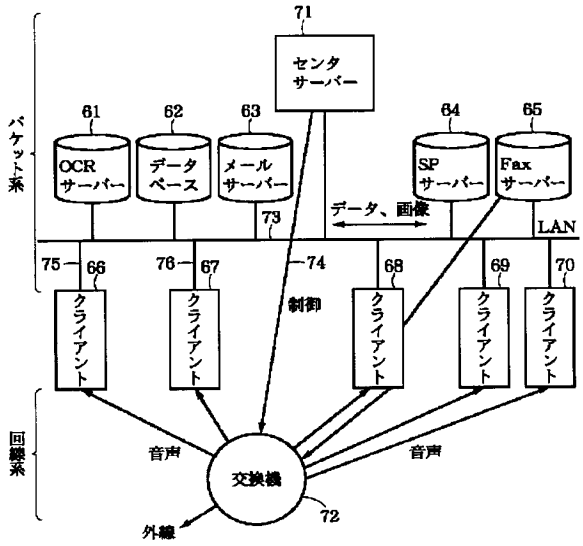


【図１１】

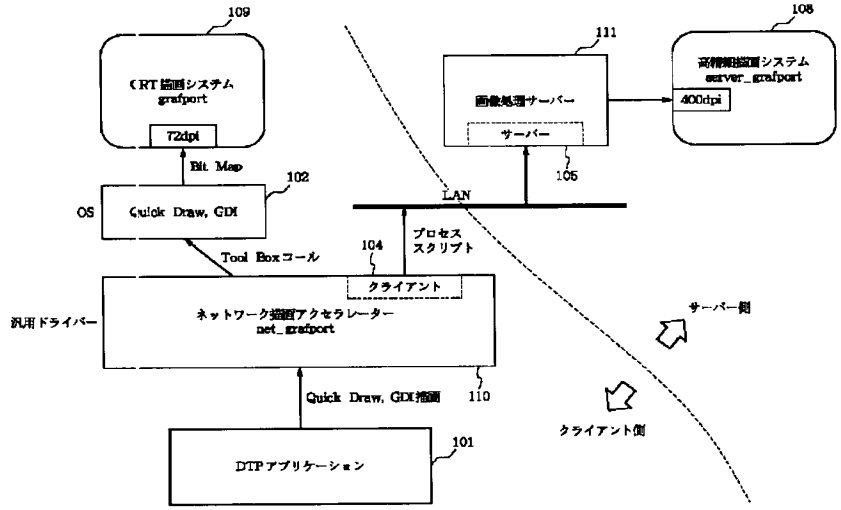
アプリ　クライアント　　センタサーバー

```
MM_open ("server","fax_service") ;
        Message ("MM_open server fax_service")---→
                                    ←--- Message ("service_id MM_
                                         open_ack")
        gservice_id = service_id ;
        return (MM_open_ack) ;

MM_send (service_id,"MM_send"," G3"," data"," destination") ;
        Message ("service_id MM_send G3 data destination")-→
                                    ←--- Message ("service_id MM_
                                         send_ack")
        return (MM_open_ack) ;

MM_data (service_id,"MM_send"," data_id"," length"," content_
        buffer") ;
        Message ("service_id MM_send data_id length
        content") ---→
                                    ←--- Message ("service_id MM_
                                         data_ack data_id")
        gdata_id = data_id ;
        return (MM_data_ack) ;
                                    :
                                    :

MM_close (service_id) ;
        Message ("service_id MM_close")---→
                                    ←--- Message ("service_id MM_
                                         close_ack")
        return (MM_close_ack) ;
```

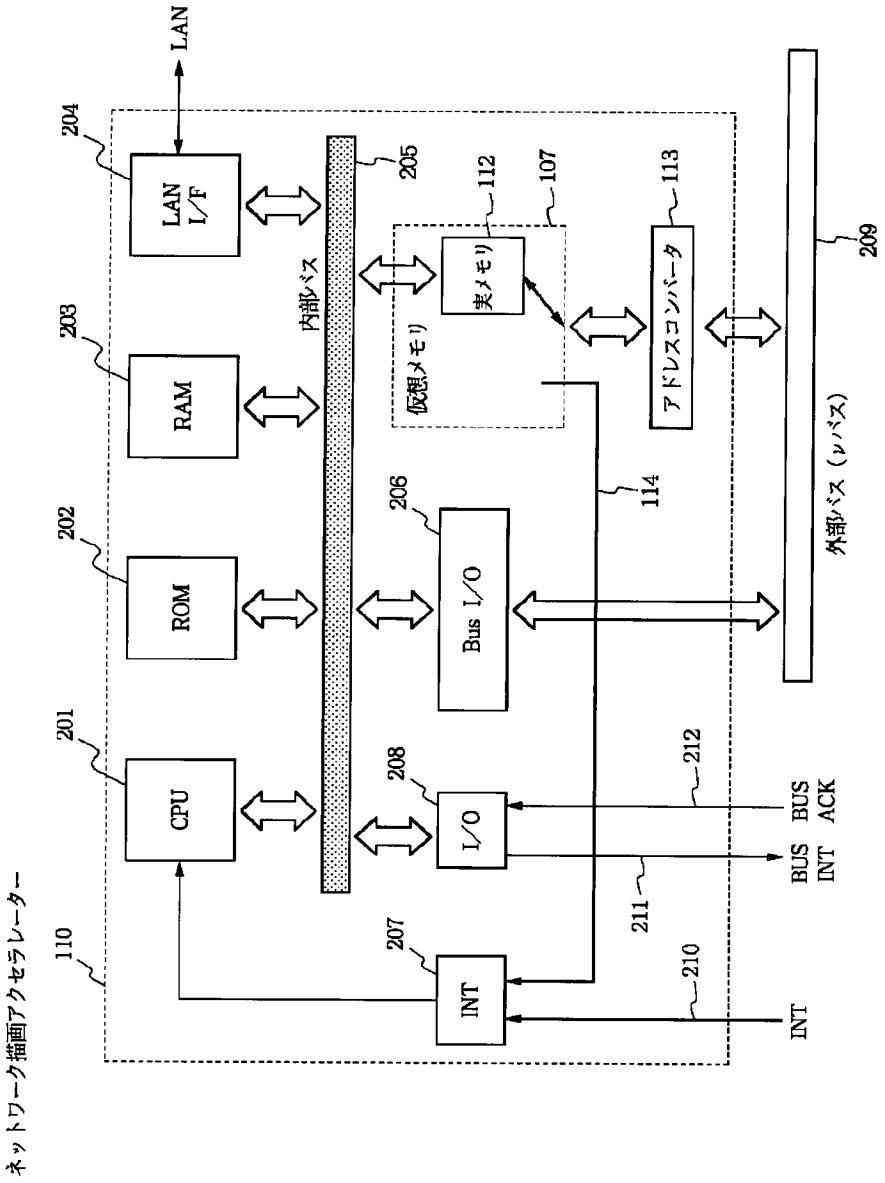
【図１２】



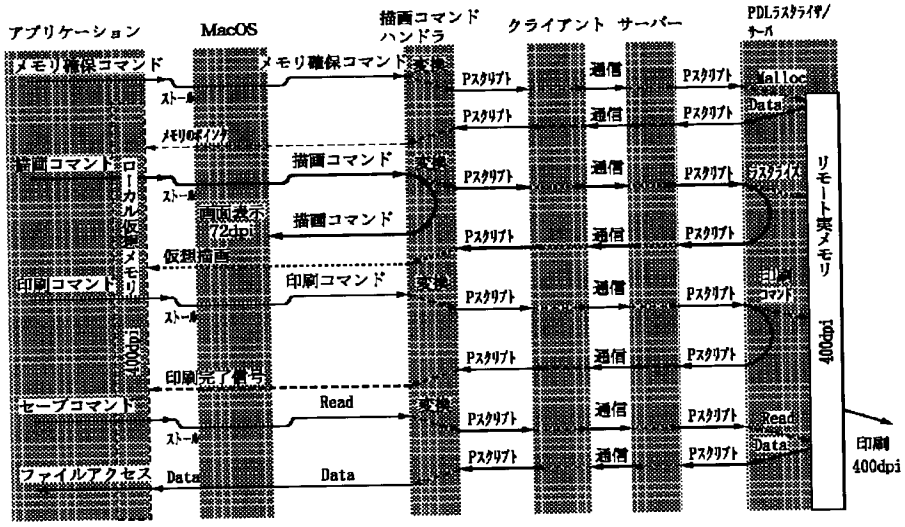
【図１４】



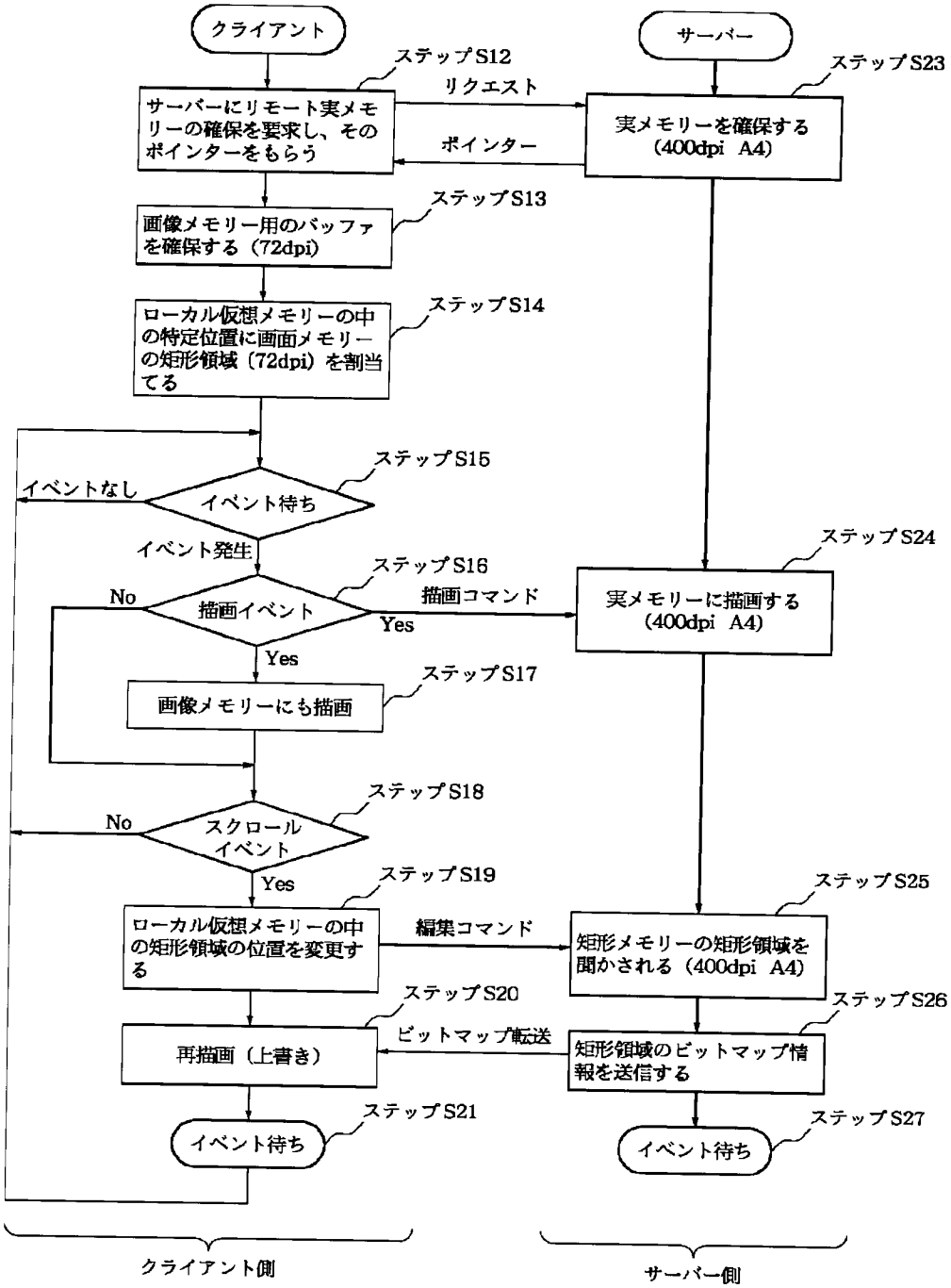
【図１５】


ネットワーク描画アクセラレーター
110
CPU
201
ROM
202
RAM
203
LAN I/F
204
LAN
内部バス
205
仮想メモリ
実メモリ
112
107
アドレスコンバータ
113
外部バス（システムバス）
209
Bus I/O
206
I/O
208
INT
207
114
211
212
210
BUS INT
BUS ACK
INT

【図１６】


アプリケーション
MacOS
描画コマンド
ハンドラ
クライアント
サーバー
PDLラスタライザ/
サーバ
メモリ確保コマンド
ストーム
メモリのポインタ
描画コマンド
ローカル仮想メモリ
画面表示
72dpi
仮想描画
印刷コマンド
印刷完了信号
セーブコマンド
Read
ファイルアクセス
Data
変換
Pスクリプト
通信
Malloc
Data
ラスタライズ
リモート実メモリ
印刷
コマンド
Read
Data
印刷
400dpi

【図１７】


クライアント
サーバー
ステップS12
ステップS23
サーバーにリモート実メモリーの確保を要求し、そのポインターをもらう
リクエスト
ポインター
実メモリーを確保する (400dpi A4)
ステップS13
画像メモリー用のバッファを確保する（72dpi）
ステップS14
ローカル仮想メモリーの中の特定位置に画面メモリーの矩形領域〔72dpi〕を割当てる
ステップS15
イベントなし
イベント待ち
イベント発生
ステップS24
ステップS16
No
描画イベント
描画コマンド
Yes
実メモリーに描画する (400dpi A4)
Yes
ステップS17
画像メモリーにも描画
ステップS18
No
スクロールイベント
Yes
ステップS19
ステップS25
ローカル仮想メモリーの中の矩形領域の位置を変更する
編集コマンド
矩形メモリーの矩形領域を聞かされる（400dpi A4）
ステップS20
ステップS26
再描画（上書き）
ビットマップ転送
矩形領域のビットマップ情報を送信する
ステップS21
ステップS27
イベント待ち
イベント待ち
クライアント側
サーバー側